

جامعة طرابلس
كلية تقنية المعلومات



Advanced Database قواعد البيانات المتقدمة
ITSE312



أستاذ المادة – حسن علي حسن

المحاضرة الحادية عشر

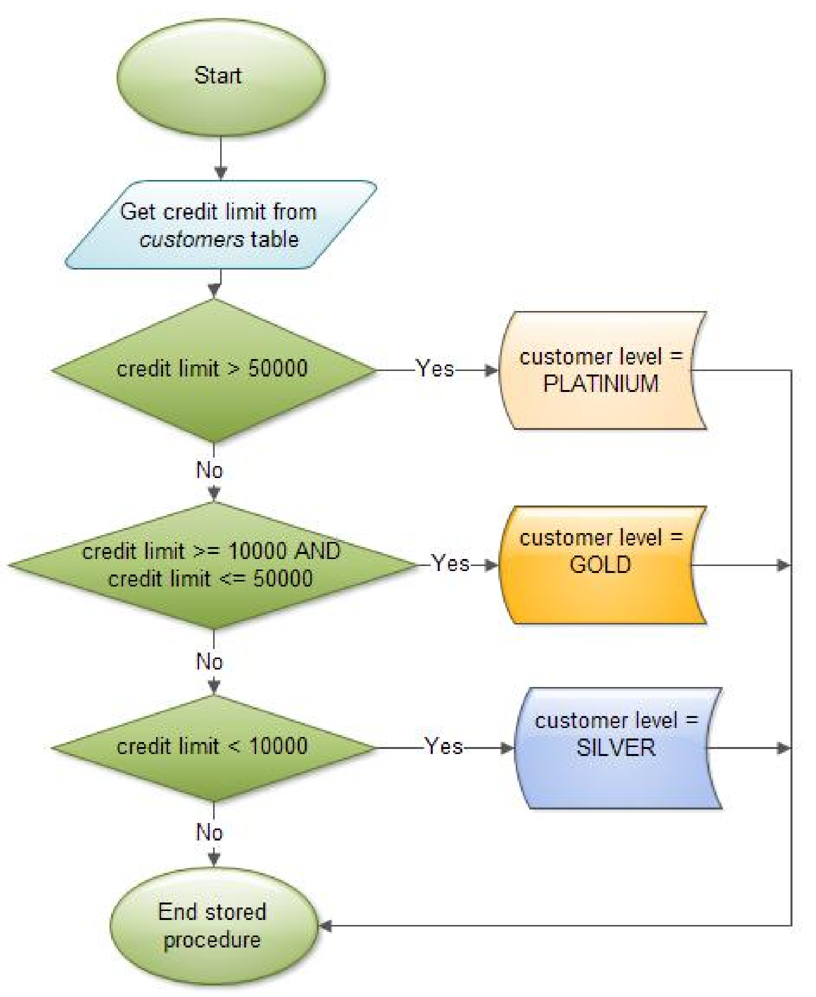


MySQL IF statement:

- **تستخدم جملة IF** لتنفيذ مقطع معين من جمل sql بناء على شرط معين.
- التركيبة العامة لها :

```
IF if_expression THEN commands
   [ELSEIF elseif_expression THEN commands]
   [ELSE commands]
   END IF;
```

- إذا تحقق شرط *if_expression* سيتم تنفيذ **commands** ، إذا لم يتحقق الشرط سينتقل الى *elseif_expression* ، وإذا لم يتحقق إي شرط سيتم تنفيذ ما بعد ELSE.



- **MySQL IF statement:**
- **مثال على جملة IF:** سنقوم بتمرير رقم الزبون customer number الى stored procedure لنتحصل على credit limit من الاستفسار، ثم نستخدم جملة IF لمقارنة قيمة credit limit لترجيع نص معين.
- إذا كانت قيمة credit limit اكبر من 50000 ترجع PLATINUM ، وإذا كانت اصغر من 50000 واكبر من 10000 ترجع GOLD، وإذا كانت اصغر من 10000 ترجع SILVER.



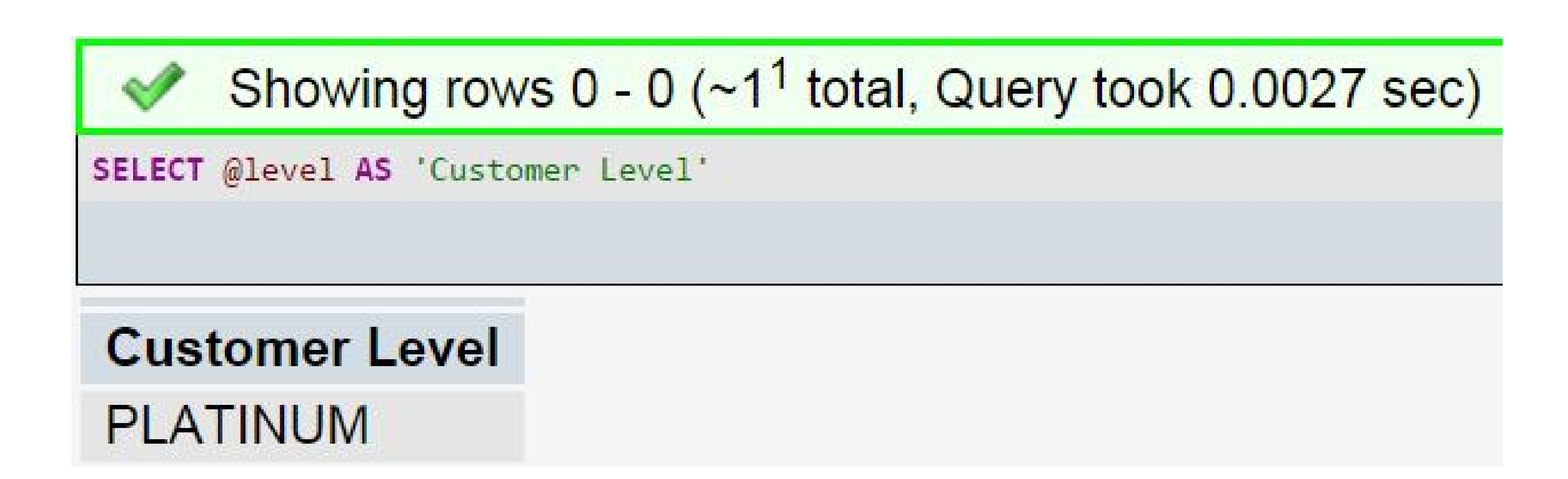
- :MySQL IF statement
- مثال على جملة IF:

```
DELIMITER $$
CREATE PROCEDURE GetCustomerLevel( IN p_customerNumber int(11), OUT p_customerLevel varchar(10))
BEGIN
    DECLARE creditlim double;
    SELECT creditlimit INTO creditlim FROM customers
    WHERE customerNumber = p_customerNumber;
    IF creditlim > 50000 THEN SET p_customerLevel = 'PLATINUM';
    ELSEIF (creditlim <= 50000 AND creditlim >= 10000) THEN
        SET p_customerLevel = 'GOLD';
    ELSEIF creditlim < 10000 THEN SET p_customerLevel = 'SILVER';
    END IF;
END$$
```



- **MySQL IF statement:**
- نستدعي stored procedure مع تمرير البارمترات كالآتي:

- CALL GetCustomerLevel(112,@level);
- SELECT @level AS 'Customer Level';





- القوادح أو النوابض أو التريقر **MySQL Triggers:**
- **Trigger** عبارة عن برنامج مخزن في قاعدة البيانات و مرتبطة بجدول، يتم تنفيذه بشكل آلي عند حدوث عملية معينة مثل insert, update or delete.
- **Trigger** يشابه stored Procedure إلا أنه لا يتم تنفيذه بإستدعاء إسمه إنما يتم تنفيذه في حالة إجراء عملية insert, update or delete على جدول معين بشكل آلي.



القوادح أو النوابض MySQL Triggers:

- الصيغة العامة

```
CREATE TRIGGER trigger_name trigger_time trigger_event
ON tbl_name
FOR EACH ROW
BEGIN
...
END
```

حيث *trigger_name* هو *إسم التريقر*

و *trigger_time* هو وقت التريقر و يحدد متى يتم تنفيذ التريقر ويكون هذا إما قبل (BEFORE) او بعد (AFTER) العملية

و *trigger_event* يحدد نوع العملية التي يتم فيها تنفيذ التريقر وياخذ أحد القيم الثلاث INSERT أو UPDATE أو DELETE

و *tbl_name* يحدد إسم الجدول الذي يتعلق به التريقر



القوادح أو النوابض MySQL Triggers:

يمكننا استخدام الامران OLD و NEW للوصول الى قيم الحقول قبل وبعد التغيير.

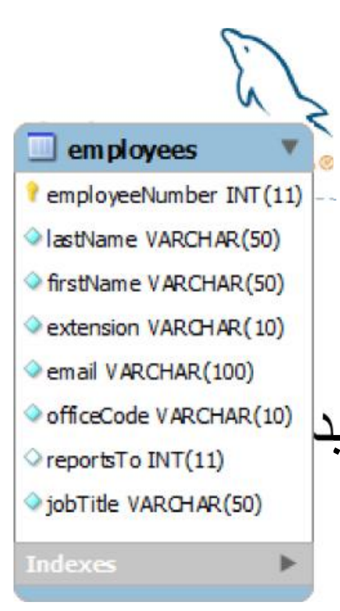
يتم استخدام الامر OLD.column_name للوصول الى قيمة الحقل قبل الغاءه او تعديله

يتم استخدام الامر NEW.colum_name للوصول الى القيمة الجديدة التي تم اضافتها او تعديلها

في التريقر الذي يحدث عند اجراء علمية الادخال INSERT يمكننا فقط استخدام الامر NEW.column_name للوصول الى القيمة المدخلة للحقل وذلك لعدم وجود اي قيمة جديدة.

في التريقر الذي يحدث عن عملية التعديل يمكننا ان نستخدم الامر OLD.column_name للحصول على قيمة الحقل قبل التعديل والامر NEW.column_name للحصول على قيمة الحقل بعد التعديل

في التريقر الذي يحدث عند اجراء عملية الالغاء DELETE يمكننا فقط استخدام الأمر OLD.column_name للوصول الى القيمة القديمة للحقل وذلك لعدم وجود اي قيمة جديدة.

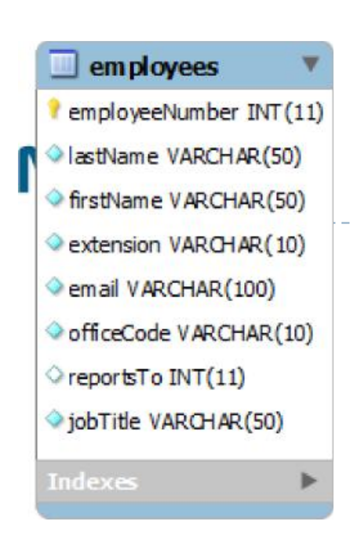


التعامل مع الجداول في MYSQL

- القوادح أو النوابض **MySQL Triggers:**
- **مثال:** لدينا جدول الموظفين employees وسنقوم بتكوين جدول جديد

باسم employees_audit لتخزين البيانات قبل البدء في تشغيل التريقر.

```
CREATE TABLE employees_audit (
    id int(11) NOT NULL AUTO_INCREMENT,
    employeeNumber int(11) NOT NULL,
    lastname varchar(50) NOT NULL,
    changedon datetime DEFAULT NULL,
    action varchar(50) DEFAULT NULL,
    PRIMARY KEY (id) )
```



التعامل مع الجداول في MYSQL

- القوادح أو النوابض **MySQL Triggers:**
- **مثال:** سنقوم بإنشاء تريقر يعمل **قبل** القيام بعملية **التحديث** BEFORE UPDATE في جدول employees والمرتبط بالتريقر.

```
DELIMITER $$
CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW BEGIN
     INSERT INTO employees_audit  SET action = 'update',
     employeeNumber = OLD.employeeNumber,
        lastname = OLD.lastname,  changedon = NOW();
END $$
DELIMITER ;
```



- القوادح أو النوابض **MySQL Triggers:**
- **مثال:** سنقوم بعرض جدول employees_audit قبل إجراء عملية التحديث على جدول الموظفين **employees** وبعد ، ثم نلاحظ ما يحدث.

```
mysql> show tables;
+----------------------------+
| Tables_in_classicmodels |
+----------------------------+
| customers               |
| employees               |
| employees_audit         |
| offices                 |
| orderdetails            |
| orders                  |
| orders3                 |
| payments                |
| productlines            |
| products                |
| tasks2                  |
+----------------------------+
11 rows in set (0.00 sec)

mysql> select * from employees_audit;
Empty set (0.00 sec)

mysql> update employees set lastname = 'hasan' where employeeNumber = 1056;
Query OK, 1 row affected (0.13 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees_audit;
+----+----------------+-----------+---------------------+--------+
| id | employeeNumber | lastname  | changedon           | action |
+----+----------------+-----------+---------------------+--------+
|  1 |           1056 | Patterson | 2015-12-22 22:28:37 | update |
+----+----------------+-----------+---------------------+--------+
1 row in set (0.00 sec)
```

```
CREATE TRIGGER before_employee_update
   BEFORE UPDATE ON employees
   FOR EACH ROW BEGIN
    INSERT INTO employees_audit  SET action = 'update',
    employeeNumber = OLD.employeeNumber,
       lastname = OLD.lastname,  changedon = NOW();
END
```



- **القوادح أو النوابض MySQL Triggers:**
- **مثال:** في المثال السابق تم تشغيل التريقر بشكل آلي عند إجراء عملية التحديث على جدول الموظفين، حيث قام التريقر بإضافة سجل جديد في جدول employees_audit بقيم الحقول القديمة من جدول الموظفين employees قبل إجراء عملية التعديل عليها.
- في المثال التالي : سنقوم بإنشاء تريقر يقوم بعملية التعديل في مرتبات الموظفين بطرح 300 د منها للموظفين الذين علاواتهم أكثر من 500 ، وذلك قبل القيام لعملية الاضافة في جدول الموظفين.



القوادح أو النوابض MySQL Triggers:

- مثال2: لدينا الجدول Emp التالي:

id الرقم	Ename الاسم	City المدينة	Designation الصفة	Salary المرتب	Perks العلاوات
1	Rahul	Delhi	Manager	10300	853
2	Gaurav	Mumbai	Assistant Manager	10300	853
3	Chandan	Banglore	Team Leader	15450	999
5	Tapan	Pune	Developer	20600	1111
6	Amar	Chennai	Developer	16000	1124
7	Santosh	Delhi	Designer	10000	865
8	Suman	Pune	Web Designer	20000	658

- التريقر

```
mysql> delimiter //
mysql> CREATE TRIGGER ins_trig BEFORE INSERT ON Emp
    -> FOR EACH ROW
    -> BEGIN
    -> UPDATE Employee SET Salary=Salary-300 WHERE Perks>500;
    -> END;
    -> //
```



القوادح أو النوابض **MySQL Triggers:**

- **مثال2:** سنقوم بعملية الاضافة على جدول emp بهذا الامر:

```
mysql> delimiter ;
mysql> INSERT INTO Emp VALUES(9,'Rajesh','Delhi','Developer',15000,658);
mysql> SELECT * FROM Emp
```

id الرقم	Ename الاسم	City المدينة	Designation الصفة	Salary المرتب	Perks العلاوات
1	Rahul	Delhi	Manager	10000	853
2	Gaurav	Mumbai	Assistant Manager	10000	853
3	Chandan	Banglore	Team Leader	15150	999
5	Tapan	Pune	Developer	20300	1111
6	Amar	Chennai	Developer	15700	1124
7	Santosh	Delhi	Designer	9700	865
8	Suman	Pune	Web Designer	19700	658
9	Suman	Delhi	Developer	15000	658

لاحظ ان كافة المرتبات نقصت بمقدار 300 وذلك لان كافة الموظفين يتمتعون بعلاوات اكبر من 500



- القوادح أو النوابض **MySQL Triggers:**
- **مثال3:** لنفترض أننا نريد إنشاء قادح يقوم بعملية التعديل على حساب الشخص بمقدار المبالغ التي يقوم بدفعها.

```
CREATE  TRIGGER checkBalance
AFTER INSERT ON payments
FOR EACH ROW
BEGIN
UPDATE accounts
set Balance = Balance - NEW.paymentAmount where AccountID=NEW.ID
END;
```



- القوادح أو النوابض MySQL Triggers:
- مثال4: إذا أردنا إنشاء قادح يقوم بالتقاط حدث التعديل على جدول employees ليسجل القيم قبل وبعد التعديل في الجدول Log .
- في البداية نقوم بإنشاء جدول Log ليتم فيه تخزين البيانات فيما بعد:

- create table log (id int auto_increment primary key,
- OldValue double, Newvalue double)



القوادح أو النوابض MySQL Triggers:

- **مثال4:** نقوم بإنشاء القادح بإسم empl_salary ، نكتب الصيغة التالية :

```
delimiter //
CREATE  TRIGGER empl_salary
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
INSERT INTO Log (oldValue, newValue) Values (OLD.`SALARY`, NEW.`SALARY`);
END//
delimiter ;
```



- القوادح أو النوابض **MySQL Triggers:**
- **مثال4:** إنشاء القادح:

Your SQL query has been executed successfully (Query took 0.0100 sec)

```
delimiter // CREATE TRIGGER empl_salary AFTER UPDATE ON employees FOR EACH ROW BEGIN INSERT INTO Log
(oldValue, newValue) Values (OLD.`SALARY`, NEW.`SALARY`); END // delimiter ;
```

- بيانات جدول الموظفين employees

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24000.00
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1987-06-18	AD_VP	17000.00
102	Lex	De Haan	LDEHAAN	515.123.4569	1987-06-19	AD_VP	17000.00
103	Alexander	Hunold	AHUNOLD	590.423.4567	1987-06-20	IT_PROG	9000.00
104	Bruce	Ernst	BERNST	590.423.4568	1987-06-21	IT_PROG	6000.00

القوادح أو النوابض MySQL Triggers:

- **مثال4:** جدول الموظفين employees قبل تنفيذ عملية التعديل عليه.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24000.00
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1987-06-18	AD_VP	17000.00
102	Lex	De Haan	LDEHAAN	515.123.4569	1987-06-19	AD_VP	17000.00
103	Alexander	Hunold	AHUNOLD	590.423.4567	1987-06-20	IT_PROG	9000.00
104	Bruce	Ernst	BERNST	590.423.4568	1987-06-21	IT_PROG	6000.00

- تنفيذ عملية التعديل على جدول الموظفين employees .

1 row(s) affected. (Query took 0.0372 sec)

```
UPDATE employees SET salary =30000 WHERE employee_id =100
```



- القوادح أو النوابض MySQL Triggers:

- مثال4: جدول الموظفين employees بعد تنفيذ عملية التعديل عليه.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	30000.00
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1987-06-18	AD_VP	17000.00
102	Lex	De Haan	LDEHAAN	515.123.4569	1987-06-19	AD_VP	17000.00
103	Alexander	Hunold	AHUNOLD	590.423.4567	1987-06-20	IT_PROG	9000.00
104	Bruce	Ernst	BERNST	590.423.4568	1987-06-21	IT_PROG	6000.00
105	David	Austin	DAUSTIN	590.423.4569	1987-06-22	IT_PROG	4800.00

- بيانات جدول log بعد تنفيذ عملية التعديل على جدول الموظفين.

id	OldValue	Newvalue
1	24000	30000



- **القوادح أو النوابض MySQL Triggers:**
- لمعرفة **Triggers** الموجودة في قواعد البيانات نستخدم الصيغة التالية:
- **SHOW TRIGGERS**;
- لمعرفة **Trigger** موجودة في قاعدة بيانات معينة نستخدم الصيغة التالية:
- **SHOW TRIGGERS FROM** sampledb;
- لمعرفة **Trigger** موجودة في قاعدة بيانات معينة وفي جدول معين نستخدم الصيغة التالية:
- **SHOW TRIGGERS FROM** classicmodels **WHERE** \`**table**\` = 'products';



- القوادح أو النوابض **MySQL Triggers:**

Trigger	Event	Table	Statement	Timing
before_employee_update	UPDATE	employees	BEGIN INSERT INTO employees_audit SET...	BEFORE
before_products_update	UPDATE	products	BEGIN INSERT INTO price_logs(product_code,price) V...	BEFORE

- لحذف **Trigger** موجودة في جدول معين في قاعدة البيانات نستخدم الصيغة التالية:

- **DROP TRIGGER** table_name.trigger_name

- DROP TRIGGER employees.before_employees_update

نهاية المحاضرة





Database Schema

productlines
productLine VARCHAR(50)
textDescription VARCHAR(4000)
htmlDescription MEDIUMTEXT
image MEDIUMBLOB
Indexes

products
productCode VARCHAR(15)
productName VARCHAR(70)
productLine VARCHAR(50)
productScale VARCHAR(10)
productVendor VARCHAR(50)
productDescription TEXT
quantityInStock SMALLINT(6)
buyPrice DOUBLE
MSRP DOUBLE
Indexes

orderdetails
orderNumber INT(11)
productCode VARCHAR(15)
quantityOrdered INT(11)
priceEach DOUBLE
orderLineNumber SMALLINT(6)
Indexes

employees
employeeNumber INT(11)
lastName VARCHAR(50)
firstName VARCHAR(50)
extension VARCHAR(10)
email VARCHAR(100)
officeCode VARCHAR(10)
reportsTo INT(11)
jobTitle VARCHAR(50)
Indexes

customers
customerNumber INT(11)
customerName VARCHAR(50)
contactLastName VARCHAR(50)
contactFirstName VARCHAR(50)
phone VARCHAR(50)
addressLine1 VARCHAR(50)
addressLine2 VARCHAR(50)
city VARCHAR(50)
state VARCHAR(50)
postalCode VARCHAR(15)
country VARCHAR(50)
salesRepEmployeeNumber INT(11)
creditLimit DOUBLE
Indexes

orders
orderNumber INT(11)
orderDate DATETIME
requiredDate DATETIME
shippedDate DATETIME
status VARCHAR(15)
comments TEXT
customerNumber INT(11)
Indexes

offices
officeCode VARCHAR(10)
city VARCHAR(50)
phone VARCHAR(50)
addressLine1 VARCHAR(50)
addressLine2 VARCHAR(50)
state VARCHAR(50)
country VARCHAR(50)
postalCode VARCHAR(15)
territory VARCHAR(10)
Indexes

payments
customerNumber INT(11)
checkNumber VARCHAR(50)
paymentDate DATETIME
amount DOUBLE
Indexes